TUBITAK-UZAY-MultiCommSim – Requirements Document

1. Project Description

TUBITAK-UZAY-MultiCommSim (Multi Communication Simulation), TÜBİTAK UZAY'da kullanılan uzay simülasyon cihazı üzerinde geliştirilen bir sistemdir. Bu sistemde amaç, **yalnızca tek bir IP ve tek bir port** üzerinden **birden fazla client-server uygulamasının** birbirinden bağımsız, yalıtılmış biçimde iletişim kurmasını sağlamaktır.
Projede **server bileşenleri mutlaka Docker container** içinde çalışmalıdır. Client bileşenleri Docker içinde olmak **zorunda değildir**, dışarıdan bağlantı kurabilirler. Port çoklama veya dinamik port atama çözümleri yalnızca yedek (B planı) olarak düşünülmektedir. Asıl amaç, **port paylaşımı üzerinden bağlantı yönlendirme ve ayrıştırma mantığı geliştirmektir**.

2. Project Goals

The project aims to achieve the following objectives:

- Tek bir IP ve port üzerinden çoklu client-server bağlantısı sağlamak.
- Server uygulamalarını Docker container içinde çalıştırmak.
- Client uygulamalarının Docker içinde olmasını opsiyonel kılmak.
- Port çoklama ve dinamik port atamayı yalnızca alternatif çözüm olarak tutmak.
- Veri iletişimini izole, karışmasız ve güvenli şekilde gerçekleştirmek.
- Server tarafında gelen trafiği oturumlara ayırmak için yönlendirme/ayrıştırma katmanı tasarlamak.

3. Requirements

3.1 Functional Requirements

- **FR-1. Tek IP ve Tek Port Üzerinden Çoklu Bağlantı**

Sistem, sadece bir adet IP adresi ve bir adet port (örneğin 6003) üzerinden birden fazla client-server uygulamasının eş zamanlı veri alışverişi yapabilmesini sağlamalıdır.

- **FR-2. Docker İçinde Server Zorunluluğu**

Server bileşenleri mutlaka Docker container içinde çalışmalı ve her biri izole ortamda konumlandırılmalıdır.

- **FR-3. Client Docker Bağımsızlığı**

Client bileşenleri Docker içinde olmak zorunda değildir; harici uygulama veya ağdan doğrudan bağlanabilir.

- **FR-4. Docker Üzerinden Yönetim**

Docker CLI veya Docker Compose aracılığıyla tüm yapı otomatik olarak ayağa kaldırılmalıdır.

- **FR-5. Veri Alışverişi**

Client'lar server'a veri gönderip cevap alabilecek; TCP protokolü kullanılacaktır. JSON tabanlı veri formatı tercih edilecektir.

- **FR-6. Bağlantı İzolasyonu**

Her client-server bağlantısı diğer bağlantılardan mantıksal olarak izole edilmelidir. Aynı port kullanılmasına rağmen veri karışıklığı olmamalıdır.

- **FR-7. Oturum Yönlendirme ve Ayrıştırma**

Server tarafında bir yönlendirme katmanı, gelen verileri oturumlara göre ayrıştırmalı ve ilgili container'a iletmelidir. Aynı port üzerinden gelen farklı oturumlar eş zamanlı olarak işlenmelidir.

- **FR-8. Çoklu Bağlantı Performansı**

Aynı anda en az 10 ayrı client-server çifti kesintisiz çalışabilmelidir.

- **FR-9. Çoklu Bağlantı Performansı**

Sistem öncelikle tek port üzerinden çoklu iletişimi hedeflemelidir. Port çoklama ve dinamik port atama yalnızca B planı olarak düşünülmelidir.

- **FR-10. Loglama ve İzleme**

Her bir container'ın çalışma durumu ve bağlantı logları sistem tarafından tutulmalıdır.

3.2 Data Requirements

- **Bağlantı Verileri:** Her bir client-server eşleşmesine karşılık gelen port bilgileri (dinamik veya statik). Veri alışverişi logları (timestamp, içerik, bağlantı ID), IP, bağlantı ID, client kimliği (örneğin token, ID vs.), payload içeriği.
- **Docker Konfigürasyonları:** Dockerfile tanımları. docker-compose.yml dosyaları, (Her server için Dockerfile, Docker Compose tanımları.)
- **Test Verileri:** JSON formatında dummy veri örnekleri, komut paketleri.

3.3 System Requirements

1. Technological Requirements:

- Docker 24+
- Docker Compose v2+
- Ubuntu tabanlı işletim sistemi
- Geliştirme dili: Python 3.10+ veya Java 17+
- Ağ yapılandırmaları için iptables veya benzeri firewall ayarları.

2. Hardware Requirements:

- İşlemci: 4+ çekirdekli CPU
- Bellek: Minimum 8 GB RAM
- Depolama: En az 20 GB boş alan
- Ağ: Sabit bir IP adresi atanmış cihaz, dış erişime kapalı

3.4 User Requirements

- **Geliştirici/Stajyer:** Docker ortamını hazırlar, container yapılarını oluşturur. Port yapılandırmalarını ve testlerini yapar.
- **Ağ Yöneticisi:** IP adresini yapılandırır, ağ güvenliğini sağlar. Port çakışmalarını izler.
- **Simülasyon Operatörü:** Simülasyon başlatıldığında kaç bağlantı olacağını belirler. Sistem üzerinden bağlantı durumlarını takip eder.

3.5 Performance Requirements

- **Response Time:** Bir client-server çifti arasında ortalama tepki süresi 50 ms altında olmalıdır.
- **Eşzamanlı Oturum:** Minimum 10 adet bağımsız bağlantı aynı anda çalışabilir olmalıdır.
- **Kaynak Kullanımı:** Her container başına maksimum 512 MB RAM kullanımına izin verilmelidir. Sistem kaynakları sınır aşımlarında loglanmalıdır.

4. Project Timeline

Phase	Duration	Objectives
Docker Ortamı Kurulumu	2 gün	Docker, ağ yapılandırması ve test ortamı kurulumu
Container Mimarisi Tasarımı	3 gün	Dockerfile ve compose dosyalarının hazırlanması
İletişim ve Port Yönetimi Geliştirme	4 gün	Port optimizasyonu, bağlantı izolasyonu
Test Süreci	2 gün	5-10 bağlantı ile eş zamanlı testler
Loglama ve İzleme Eklenmesi	1 gün	Log sistemlerinin kurulması

5. Risks and Mitigations

- ***Port Yönlendirme Karmaşası***
  Çözüm: Her bağlantıyı ayrıştıracak bir başlık yapısı (örneğin client ID) ile gelen veri ayrıştırılır.
- **Docker Ağ Problemleri**
  Çözüm: Docker bridge veya overlay network yapılandırması yapılır.
- ***Yüksek Sistem Kaynak Kullanımı***
  Çözüm: Her container için kaynak sınırları tanımlanmalı (Docker resource limits).
- ***Client dışarıdaysa güvenlik açığı***
  Çözüm: Kimlik doğrulama (örneğin token, bağlantı anahtarı) uygulanır.

6. Approval and Signature

Project Manager: [Yetkili Mühendis Adı – TÜBİTAK UZAY]

Student Intern: Ediz Arkın Kobak

Date: 01.07.2025

Signature: _____